A

Dissertation

on

Design and Implementation of Artificial Neural Network Using Vedic Mathematics

Submitted in

partial fulfilment of the requirement

for the award of the degree of

Master of Technology

in

VLSI Design and Embedded System

Submitted
by

Anshika
Roll No. 2K13/VLS/01

Under the Guidance of

Dr. Nidhi Goel
Assistant Professor,
Department of Electronics and Communication Engineering
Delhi Technological University

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

2013-2015

CHAPTER 1

INTRODUCTION

Artificial neural networks (ANN) play a major role in applications involving DSP. A few to mention include image processing, pattern recognition, GPS systems, speech, vision, and control systems [1-4]. ANN is similar to the human brain and can solve problems which are difficult for the conventional computers, as they can be trained just like the human brain which is not possible for the conventional computers. Additionally, they have attractive properties like adaptiveness, self-organization, nonlinear network processing and parallel processing. This has lead to the use of neural network in applications involving classification, association, decision-making and reasoning [5]. Artificial neural networks consist of massively parallel network and require parallel architecture for high speed operations in real time applications [6]. Also, Neurocomputers and neuro-computing has always been a fascinated topic of research from early 80s and 90s. A lot of research has been done on design and implementation of neural networks and hardware neuro computers [7, 8, 9, 10]. But that time almost all the researches proved unsuccessful in proving the wide use of neural network. But after a long period, literature shows that the concept of neural network and various applications associated with it made a sort of comeback [11, 12, 13]. The main hurdle that was found was ASIC implementation and hence in the last few years, a great revolution has been witnessed in this field due to the use of FPGA which are considered as the best choice for modern digital system designing. Hence, while considering these features neural networks are considered to be best suited for the VLSI technology. The possibility of hardware realization of neural network mainly depends on how efficiently single neuron can be implemented.

ANN technology has emerged as an effective computational modeling tool and has founded extensive acceptance in modeling various complex real word problems. Although ANNs are considered as the abstractions of biological counterparts, but the basic idea behind ANNs is not the replication of the operation of biological systems, rather, the main idea was to exploit the functionality of biological neural network which can solve a large number and variety of complex problems. The main advantage or reason why ANN was an attractive topic for researchers was its non linearity along with failure and fault tolerance capability which involves high parallelism and can also handle fuzzy information. ANN finds a large range of areas in which it has been applied. Few amongst them are:

- Pattern recognition of protein structure, RNA, microscopic images and DNA.
- Biomass and growth prediction.
- Handwriting recognition.
- Number plate identification.
- Traffic control applications etc.

Since ANN covers wide range of applications, hence the speed of neural network has always been a main parameter of concern. Various efforts have been done and research is going on to improve the speed of the artificial neural network. As described before that the possibility of hardware realization of neural network mainly depends on how efficiently single neuron can be implemented. Hence, in this work, by enhancing the speed of the multiplier which is involved in almost each and every step of ANN, the overall speed of a single neuron or 'Perceptron' has been improved. This in turn will enhance the processing speed of the various applications which are based on artificial neural network.

1.1 Motivation

Multiplier is a crucial building block of several signal processing applications, as they play an important role in designing an efficient architecture [14, 15]. Recently, several high speed multipliers designed using Vedic mathematics have been reported [16, 17]. Vedic mathematics reduces the computation time in calculations and thereby making it very simple and easier one compared to the conventional mathematics. The reason behind the simplicity of Vedic formulae is that they follow the natural principles on which the human mind works. Vedic mathematics is a collection of arithmetic rules that allow more efficient speed implementation. Recently Vedic mathematics based ALU and modulators have been reported [18].Hence, the efficiency of Vedic mathematics can be utilized to design very efficient and high speed neural network too, which can be utilized for various applications.

Above all this, the speed has been a critical issue in functioning of any network. Hence in the present work the advantage of Vedic mathematics was the motivation behind the designing of high speed network.

1.2 Related Work

ANNs are nothing but structures comprising of densely connected nodes or neurons which are highly capable of performing parallel computations. In the last few years it has emerged as a revolutionary technique. It has the capability to build intelligent systems with cost effective and high speed solutions. There are various kinds of ways in which an ANN can be implemented i.e. digital, analog or hybrid and each one of it has its own advantages and disadvantages which depends on the type of network and the training algorithm applied. Literature shows that a lot of work has been done on designing of neural network and its various applications in image processing, electromagnetic simulation [19], and prediction of traffic speed [20] etc using different algorithms and techniques with a main focus on high speed networking. Today, ANN has been used in almost many applications of day to day life. Hence, speed of neural network has been an area of concern from past [21] and a lot of work has been done in this area.

Multiplier plays a key role in designing any high speed logic unit. Various multipliers based on booth algorithm and Wallace algorithms have been reported. Literature shows that a wide range of research has been done from past till present in designing of area efficient and high speed multipliers and recently various multipliers based on vedic mathematics have been reported [22].Vedic mathematics seems to be a promising technique in design of high speed multipliers. A lot of multipliers have been designed using Vedic mathematics which has different advantages over standard multipliers. Out of the 16 sutras of Vedic mathematics, ‘Urdhva Tiryagbhyam’ sutra is generally used for efficient designing of Vedic multiplier. The multipliers based on Vedic mathematics took over all other multipliers due to their high speed and low power capabilities. Recently Vedic multipliers designed using compressors with Urdhva Tiryagbhyam sutra have proved to be the most promising one in terms of power, area and delay. Hence, Vedic mathematics has been proved as one of the robust technique in designing various applications involving arithmetic operations. Vedic mathematics based efficient MAC unit has also been reported [23].

Logic gates and logic circuits play a major role in the theory of computation. Computation carried out by man or by machine is a physical activity, and is ultimately governed by physical principles. An important role for mathematical theories of computation is to reduce in their axioms, in a different way. With this support, the focus can be on the abstract modeling of complex computing processes without having any need to verify at every step the physical realizability of the mode [24]. Thus, a Boolean logic (using, say, the AND, NOT, and OR primitives) can be chosen with the confidence that any network designs in this way is immediately translatable into a working circuit requiring only well-understood, readily available components (the “gates”, "inverters”, and “buffers” of any suitable digital-logic family).

And since multiplication is a crucial application in designing of any neural network, so it can be used in designing high speed neural network. Until now no work has been proposed which can combine the advantages of both the techniques to design a neural network based on Vedic mathematics.

1.3 Objective and Scope of the Project

The objective of this thesis is to estimate the performance of a neuron model designed with Vedic multiplier against normal multiplier using the features of ANN & Vedic multiplier using VHDL and to verify the results using Modelsim. Also, a brief introduction to neuronal gates has been done and the hardware implications of both the schemes have been shown.

1.4 Organization of Thesis

The thesis outline is as follows:

Chapter 2: This chapter provides the literature review of the two most promising technologies i.e. Artificial Neural Network and Vedic mathematics with a detailed description of both the technologies where the objectives and the features of ANN are illustrated. Also it provides an insight to the multipliers where the introduction to multipliers and Vedic mathematics with Urdhva Tiryagbhyam sutra, along with its algorithm is presented.

Chapter 3: This chapter gives an approach to design and analysis of Neuronal logic which explains briefly about various neuronal Logic gates. Also the various implications in designing of few neuronal logic gates have been explored.

Chapter 4: This chapter deals with the proposed work i.e. the digital design of the neural network based on Vedic mathematics has been presented along with the need and advantages of the proposed work.

Chapter 5: In this chapter all the simulation results obtained from the proposed structure has been shown.

Chapter 6: This chapter deals with the conclusion from the interpreted results and explores the future work for this technology.

CHAPTER 2

LITERATURE REVIEW

The human brain comprises of numerous small cells known neurons which form a network to perform the brain activities. As the neural network is highly nonlinear and has a parallel architecture it performs tasks in a fraction of second. Because of this parallel architecture, the computation speed of a human brain is much faster than a conventional machine. [4].The computation can be viewed as a system in which the inputs are received from an external stimulus. The receptors convert the external stimulus into electrical impulses that convey the information to the neural net and depending on the electrical impulses the neural net conveys or transmits the information to effectors to provide an optimal response to stimulus as shown in Fig.1. The brain is considered to be composed of an integral constituent known as neuron. The brain learns to distinguish different patterns by training these neurons which give the appropriate output [25].
In order to emulate the brain, the concept of artificial neural network has been designed to model the way in which the brain performs a particular task or function of interest.

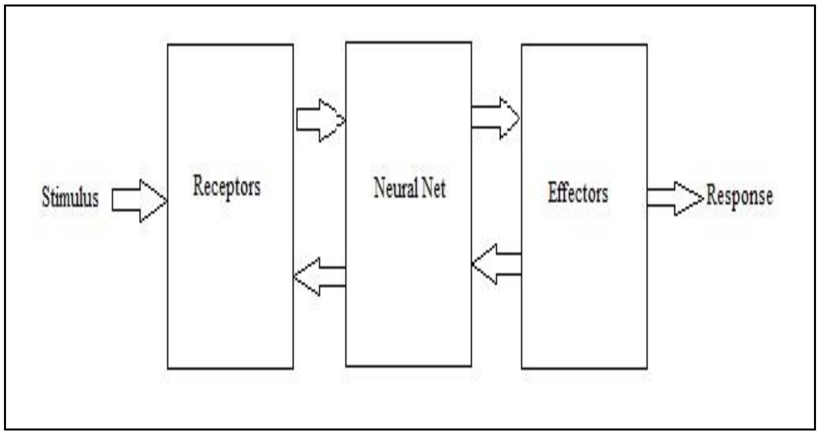

Fig.1 Block diagram representation of nervous system.

A neural network is a massively parallel distributed network made up of simple processing units. These processing units have a natural tendency of storing experiential knowledge and making it available for use [26]. It is analogous to the brain in two aspects:

- The network obtains knowledge from its environment through a learning process.
- The knowledge that has been acquired is stored in interneuron connection strengths known as synaptic weights.

Multiplication is an essential and basic function in arithmetic procedures and Vedic mathematics is an ancient methodology developed by Indians, which is capable of faster intellectual computation. Multiplication is one of the most important operations in digital neural network based systems. There are several Multiplication-based operations. Frequently used such Computation- Intensive Arithmetic Functions are multiply and Accumulate (MAC) and inner product [27]. Further, multipliers play a major role in the overall power consumption of any system. Therefore, reducing their power dissipation satisfies the overall power budget of ANN system.

Real-time applications involving lots of arithmetic operations require a high throughput for a desired performance [28]. Multiplication is one of the major and key arithmetic operations which contribute a major part in the speed of any computation. The essential requirements for many applications developed using neural network are reduction of the time delay and power consumption.

Out of the four known renowned Vedas, Sthapatya Veda which is part of Atharva Veda describes about Vedic mathematics in detail. All the modern mathematical terms such as trigonometry, arithmetic, factorization, geometry, quadratic equation and calculus has been covered in it. Shri Shankaracharya Teerathji compiled his work and gave 16 sutras after doing an exhaustive research in Atharva Veda. It has been accepted from ages and cannot be disregarded because it is not only mathematics but it is also a logical wonder too. It contains such powerful tools, that it has emerged as a fantasy topic for research which has crossed the boundaries of India. All the different branches of Vedic mathematics are based on 16 different sutras as shown below:

- Urdhva-Tiryagbhyam – Vertically and crosswise
- Chalana Kalanabyham – Differences and Similarities.
- Vyashtisamanstih – Part and Whole.
- Shunyamanyat – If one is in ratio, the other is zero.

- Yaavadunam – Whatever the extent of its deficiency
- Ekadhikina Purvena – By one more than the previous One.
- Sopaantyadvayamantyam – The ultimate and twice the penultimate.
- Ekanyunena Purvena – By one less than the previous one.
- Shunyam Saamyasamuccaye – When the sum is the same that sum is zero.
- Gunakasamuchyah – The factors of the sum is equal to the sum of the factors.
- Shesanyankena Charamena – The remainders by the last digit.
- Nikhilam Navatashcaramam Dashatah – All from 9 and last from 10.
- Sankalana- vyavakalanabhyam – By addition and by subtraction.
- Paraavartya Yojayet – Transpose and adjust.
- Gunitasamuchyah – The product of the sum is equal to the sum of the product.
- Puranapuranabyham – By the completion or non completion.

Hence both the technologies i.e. ANN and Vedic mathematics have the potential to develop efficient digital systems. Therefore, the current work explores the possibility of combining these two technologies and hence it will enhance the applications associated with ANNs. Before going further, in the next sections, a detailed introduction to Artificial Neural Networks and Vedic multipliers have been provided.

2.1 Introduction to Neural Networks

The extremely high capability of neural networks which can solve those problems which cannot be modeled mathematically makes them highly efficient. They are nothing but a model which can mimic the biological neuron structures.

2.1.1 Biological Neuron

The human nervous system consists of trillions of neurons which have different types and lengths according to their body locations. Fig.2 shows the biological neuron which consists of three major functional units called axon, dendrites and cell body. The cell body consists of nucleus which has information about heredity traits. It also consists of plasma which does the work of holding the molecular equipment used for producing the material which is needed by the neuron. The signals from the other neurons are being received by dendrites and it passes them over to the cell body. 0.25mm^2 is the approximate total area of the dendrites of a typical neuron. Synapse is the point where two neurons meet. And, the signal received from the axon by the cell body is carried through the synapse to the dendrites of the neighboring neurons. Fig.3 shows the schematic of signal transfer between two neurons through synapse.

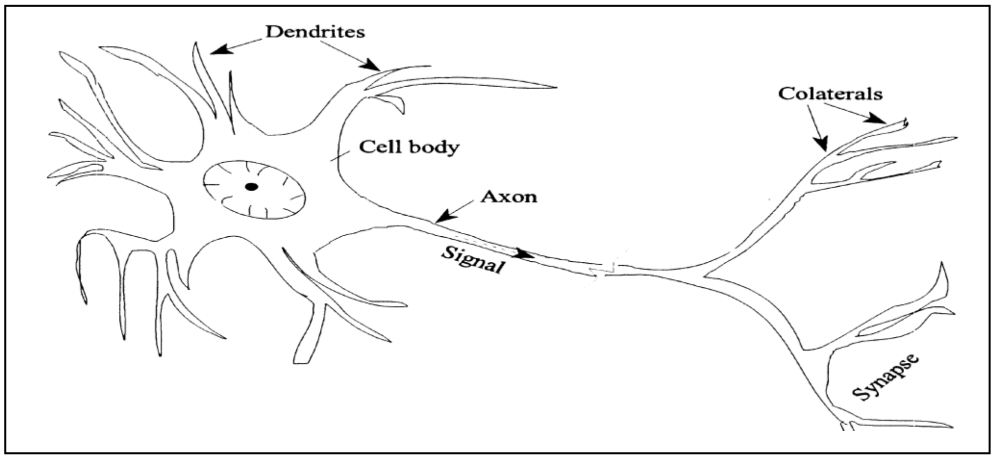

Fig.2 Schematic of biological neuron.

An impulse travels within the dendrites and through the cell body towards the pre-synaptic membrane of the synapse. A chemical generally referred as neurotransmitter is released from the vesicles as soon as it arrives at the membrane. Its quantity is proportional to the strength of the incoming signal. The neurotransmitter diffuses within the synaptic gap towards the post synaptic membrane and eventually in dendrites of neighboring neurons. And hence, according to the threshold level, it forces them to generate a new electrical signal. In the same manner the generated signal passes through other neurons. The intensity of each signal coming from each feeding neurons, their strength, and the receiving neuron’s threshold decides the amount of signal that passes through a receiving neuron. Because of the large number of dendrites/synapses in a neuron, many signals can be transferred and received simultaneously which in turn can either excite or inhibit the firing of the neuron. This mechanism was the fundamental step in the operation of building unit of ANN and other neuro computing development.

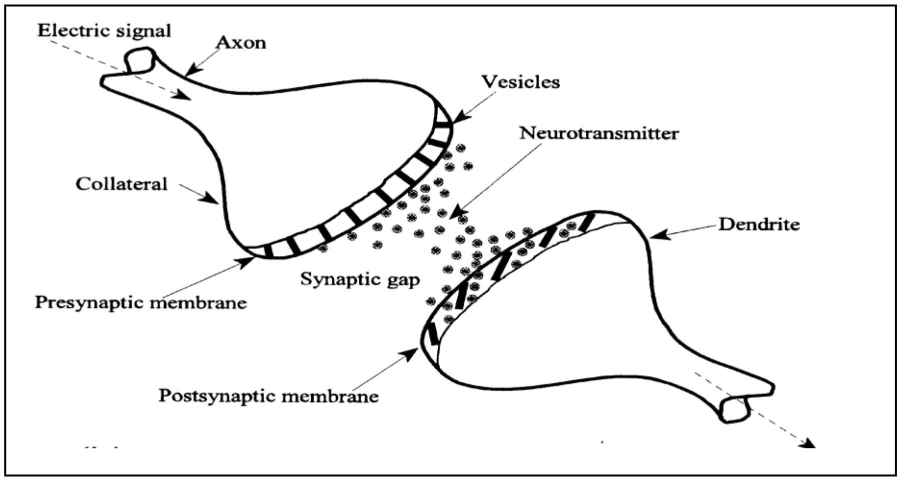

Fig.3 Mechanism of signal transfer between two biological neurons.

2.1.2 Model of Neuron

In the year 1943, researchers McCulloch and Pitts introduced the artificial neuron model after many findings with the human biological system. There are several models of neuron proposed by various scientists after this finding. One such model is known as the ‘Perceptron’ model, which was introduced by Rosenblatt in 1958 and is a single artificial neuron. Perceptron was named since the human eye perceives what it sees. As the biological neuron receives input from one of its dendrites, the artificial neuron receives its input stimuli from the surrounding environment. This input stimulus is combined with other inputs to form a net input which is then sent to an activation function which can be a linear threshold function. If the weighted sum of all the inputs is above the threshold then the output is high, which is termed as “fired”. The output of the activation function is transmitted to other adjacent neuron for firing.

The nonlinear (McCulloch) model of neuron is as shown in Fig. 4.

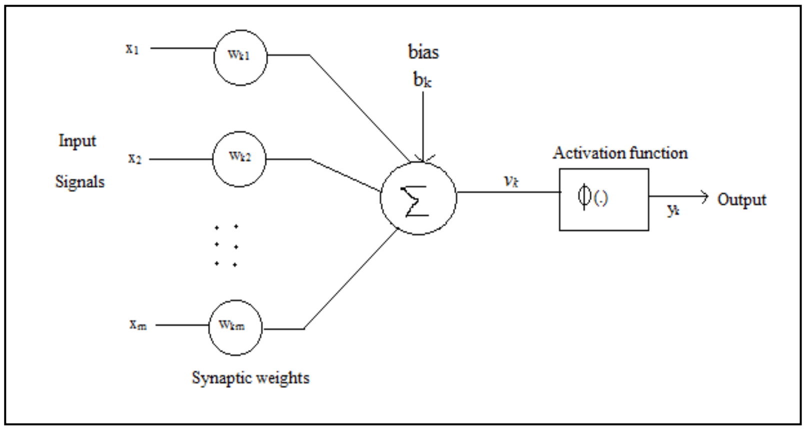

Fig.4 Nonlinear model of neuron [4].

The neuron may be described in the following way

$$u_k = \sum_{j=1}^{n} w_{kj} x_j \tag{1}$$

$$y_k = \emptyset(u_k + b_k) \tag{2}$$

where $x_1, x_2, .. x_m$ are the input signals.

w_{kj} refers to synaptic weight of neuron k.

y_k is the output of neuron k.

The relationship between induced local fields or the activation potential v_k of the neuron depends on whether bias b_k is positive or negative and is given as follows:

$$v_k = u_k + b_k \tag{3}$$

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{4}$$

$$y_k = \emptyset(v_k) \tag{5}$$

Depending on the application requirement, various activation functions could be used. Usually, three basic type of activation function i.e. threshold function, piecewise linear function and sigmoid function are used as shown in Fig.5.

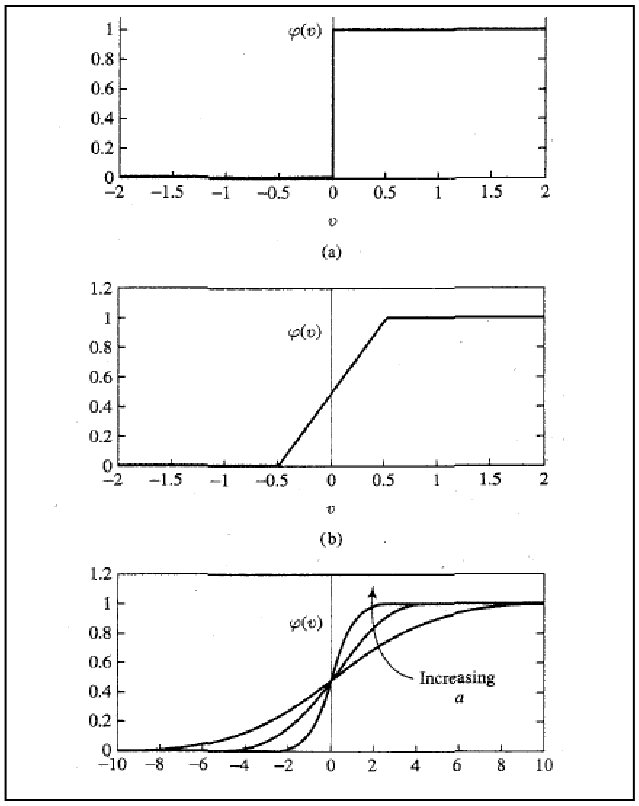

Fig.5 Activation Functions a. Threshold Function b. Piecewise linear function c. Sigmoid Function

The threshold function is given as

$$\emptyset(v) = \begin{cases} 1\ if\ v \geq 0 \\ 0\ if\ v < 0 \end{cases} \tag{6}$$

The piecewise linear function is given as

$$\emptyset(v) = \begin{cases} 1\ if\ v \geq +1/2 \\ v\ if + \frac{1}{2} > v > -\frac{1}{2} \\ 0\ if\ v \leq -\frac{1}{2} \end{cases} \tag{7}$$

The sigmoid function is given as

$$\emptyset(v) = \frac{1}{1+\exp(-av)} \tag{8}$$

where ‘a’ is the slope parameter of the sigmoid function.

In the present work, while designing neural network based on Vedic multiplier we have used the basic threshold function as the activation function.

2.2 Analogy between biological and artificial neural network

The table below shows the analogy of the components of artificial neural network with that of the biological neuron. Hence, it can be clearly seen that a human neuron can be mapped into a digital network which can work in the same way as the human brain works so as to achieve higher speed and accuracy.

TABLE I. ANALOGY BETWEEN BIOLOGICAL AND ARTIFICIAL NEURAL NETWORK.

Biological Neural Network	**Artificial neural network**
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

2.3 Topologies used in Neural Network

A major role has been played by Artificial Neural Networks (ANN) in the development of intelligent control schemes [29]. Their extensive usage is for system identification as well as controller parametrizitation. Here various types of neural networks will be discussed.

2.3.1 Feed Forward Network

The simplest form of ANN is called a Perceptron. As discussed before, it consists of a single neuron with adjustable weights and bias. In 1988 Broomhead and Lowe [30] described the design of a layered feed forward networks using radial basis function which came as an alternative to multi-layer Perceptron. A FFN is the one in which neurons of one layer are forward connected to the neurons of the other layer as shown in Fig.6. Inspite of the various variants of FFN, radial basis function networks and multi-layered networks have been used in various applications. It is because these two are considered as universal approximators. Approximators are the one using which any non linear function can be approximated with an arbitrary accuracy using a properly tuned multilayered network or a radial basis function network.

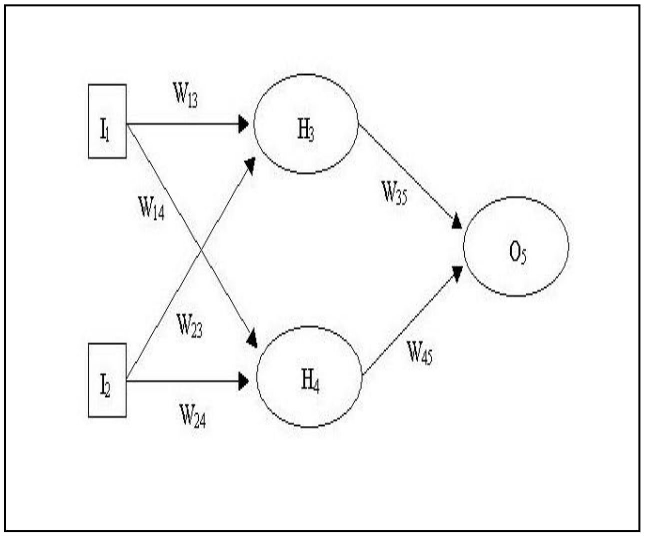

Fig.6 Architecture of feed forward network.

2.3.2 Feedback Network

The versatility of a neural network is considerably increased by the addition of feedback. As discussed before FFN gets mapped from the input space to the output. As soon as the weights get fixed the neuron is independent of the initial and past state of this neuron. Hence a feedback network is a static network. In contrast, feedback network allows feedback connection and also called as recurrent network as shown in Fig.7 .It becomes a non linear dynamic system which exhibits highly non linear dynamic behavior and thence it makes it highly interesting. Such a system has very rich temporal and spatial behaviors. These behaviors could be stable and unstable fixed points and limit cycles, and chaotic behaviors. These behaviors can be utilized to model certain cognitive functions such as unsupervised learning, associative memory, self organizing maps and temporal reasoning.

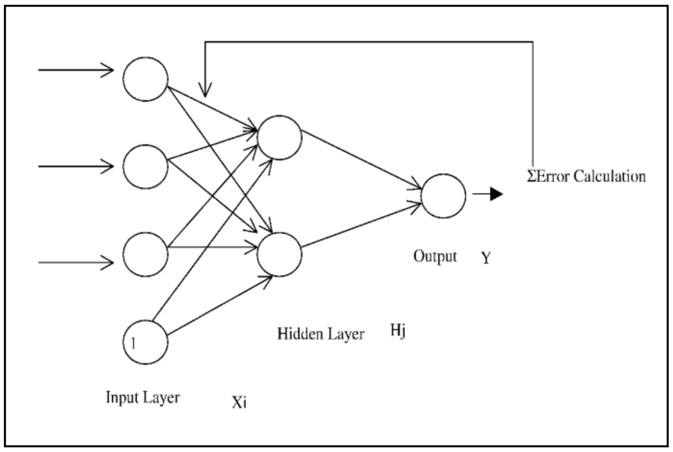

Fig.7 Architecture of feedback network.

2.3.3 Single-layer Neural Network

A single layer neural network (SNN) consists of input neurons and output neurons and is called as a single layer network because instead of considering input neurons as a layer, they are considered as a source of getting data into the network. Fig.8 clearly shows the architecture of single layer neural network.

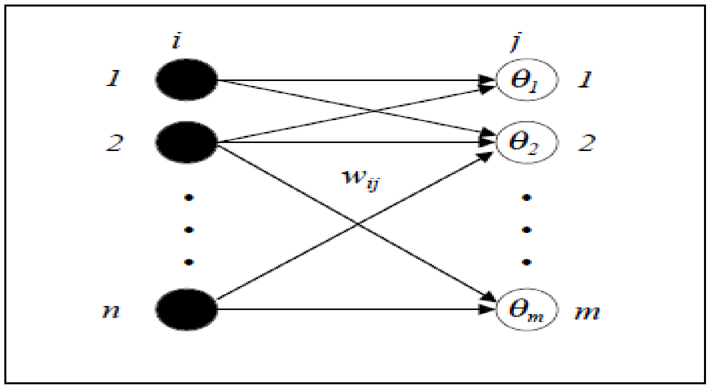

Fig.8 Architecture of single layer neural network.

2.3.4 Multi-layered Neural Network

As shown in Fig.9, a multilayer neural network (MNN) consists of an input layer of source nodes, hidden layers of neurons and an output layer of neurons. The propagation of input signal takes place on a layer to layer basis through the network in the forward direction. The response by every neuron is actuated using an activation function. They have been proved successful in solving difficult problems by training them in a supervised manner with the error back propagation algorithm.

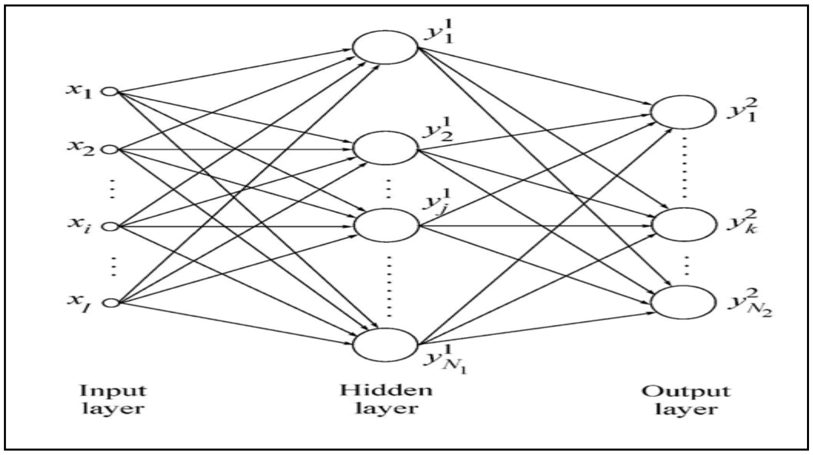

Fig.9 Architecture of a 3 layer MLP network.

2.4 Parallelism in Neural Network

To determine the best mappings and most suitable hardware structures, a careful examination of the various parallelism offered by neural networks is required. It can be of various types, for example, MIMD type or SIMD type, bit parallel or word parallel [31]. Few are explained as follows:

- Training parallelism: Here medium level (usually hundreds) of training sessions run in parallel on SIMD and MIMD processor and hence it makes them easier to fully map onto large FPGA's.

- Layer parallelism: It corresponds to multilayer networks. In such networks different layers can be processed in parallel. The level of parallelism is quiet low here. Because of this it is of limited value but can be exploited through pipelining.

- Node parallelism: This is the most important level of parallelism which corresponds to individual neurons.

- Weight parallelism: In a neural network, the inputs, weights and the products can be calculated in parallel during the computation of an output. Also, the sum of all products can be calculated with high parallelism.

- Bit level parallelism: A large variety of parallelism is available while considering the implementation level, which depends on design of individual functional unit. e.g. word parallel, bit serial, serial parallel.

The above discussion summarizes the various aspects of parallelism in context of implementation. The points are as shown below

- Depending on cost and performance rate, a tradeoff between different types of parallelism can be made.

- There is an enormous variation of parallelism available at different levels.

- Not all types of parallelism support FPGA implementation.

2.5 Overview of Back Propagation Algorithm

Fig.10 shows the generalized structure of multilayer Perceptron neural network (MLP). In the figure the neurons have been marked from 1 to N and the layers have been numbered 0 to M. The five steps which MLP uses to execute back propagation are as follows:

a) Initialization: Before training the neural network a few parameters have to be initialized and they are:

    - Synaptic weight
    - Learning rate
    - Bias of neuron

    Statistically, bias increases the chances of convergence and can be thought of as a noise which randomizes initial conditions in a better way.

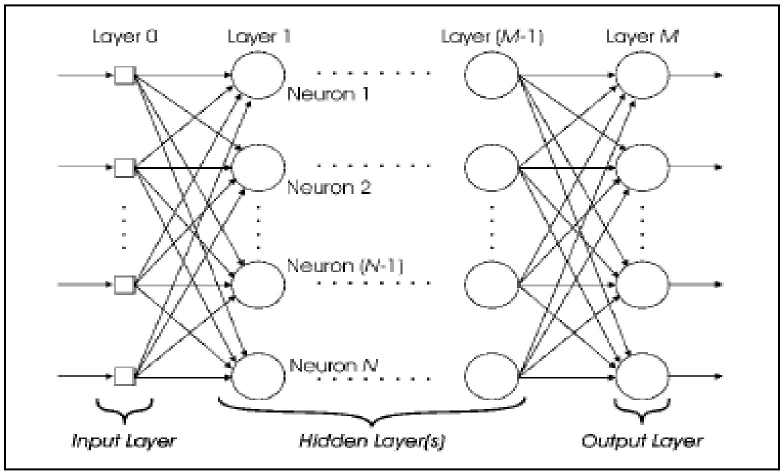

Fig.10 Generic structure of a feed forward ANN.

b) Presentation of training example: Whatever data is available, that data is presented to the network. This can be presented either individually or as a group.

c) Forward computation: As soon as forward computation starts, the available data from neuron at that time is made to propagate forward from lower layers to the higher layer via a feed forward network and hence the output is calculated.

d) Backward computation: The main aim of learning algorithm is to minimize errors. Hence in this process the updation of weights and bias takes place. The error is calculated by comparing the expected value and the actual value hat has been obtained during forward computation.

e) Iteration: In this step, iterations are reiterated for each training example in epoch until some criteria for stopping has been achieved. While running applications, MLP will compute forward computation once training gets completed.

2.6 ANN project development

The various phases of ANN project development are shown in Fig.11. The development of any ANN project requires six phases as described below:

a) Phase I: It concerns with the formation and definition of problem which is totally dependent on clear understanding of the problem which mainly includes the ‘cause-effect’ relationships.

b) Phase II: It is mainly considered as the first step in designing of actual ANN. In this, the type of ANN and a learning algorithm which fits the concerned problem is determined by the modeler. All the work related to data collection, data pre-processing, positioning of data, statistical analysis of data is also done.

c) Phase III: This phase deals with the training of data and error prediction to assess the network performance.

d) Phase IV: In this, examination for best network is done by ANN, testing against the data when training is in process, and also the comparison with other approaches is done.

e) Phase V: The hardware implementation of the network and finally the testing of embedded system are done before handling it to the user.

f) Phase VI: Updation of the developed system according to the new system variables is done.

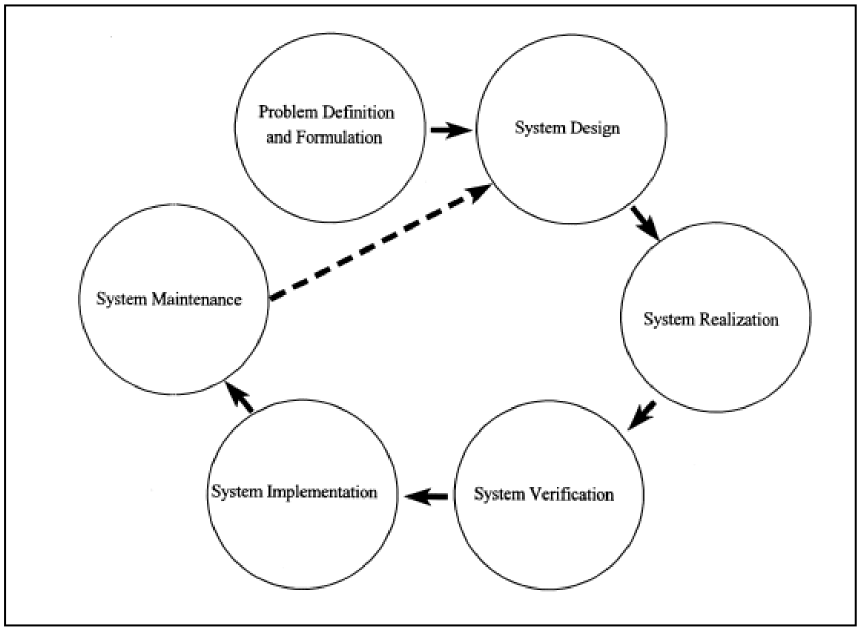

Fig.11 Various phases in an ANN development project.

2.7 Hardware Implementation of a Neuron

Since last 25 years, hardware implementations of artificial neural network have been of interest for many scientists [32]. Various hardware implementations are proposed [33]. The analog implementation uses CMOS transistors which have its own merits and demerits. The digital implementation has been proposed using FPGA by various researchers, as FPGA is the state of art technology in VLSI. [34,35,36]. The reconfigurable feature of FPGA has been a key point due to which it is preferable for ANN hardware implementation. Also FPGA is more advantageous than ASIC, as FPGAs are fully reconfigurable and cost wise is lesser than ASIC. FPGAs are more flexible with respect to hardware as well as software [37].

The implemented non-linear model of a single neuron is shown in Fig. 12.

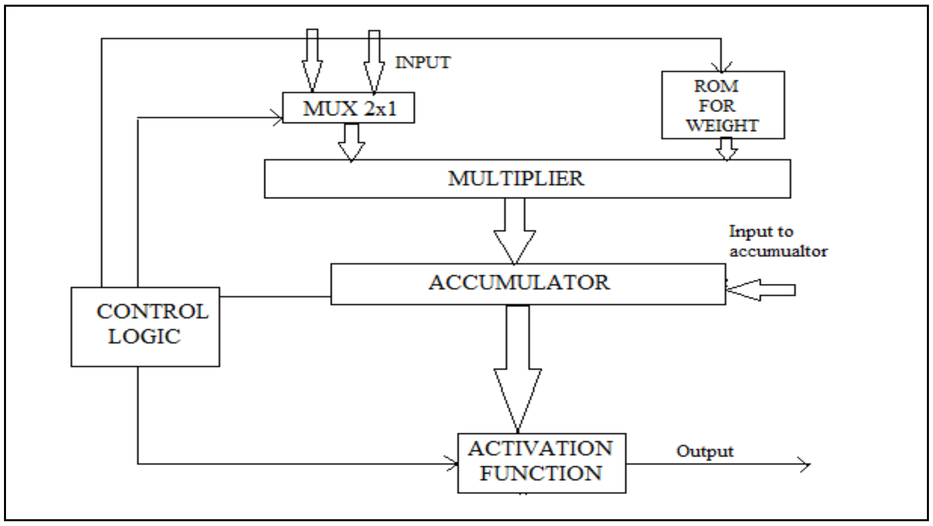

Fig.12 Block diagram of a single neuron.

It works in the same way as the biological neuron works. The input signal along with the desired weight is multiplied in a multiplier. The output of multiplier is fed to the accumulator which adds that value with that of the previous one or the other values obtained from different inputs and weights. The output of the accumulator is directed to the activation function which can be chosen as per the desired need. If the value of the accumulators output is greater than the threshold value, than the neuron is said to be fired otherwise not.

2.8 Introduction to Multiplier

The need of high speed multiplier has always been in demand because it is utilized in almost each and every application. History suggests that earlier the multiplication operation involved addition, subtraction and shifting operations. The multiplier is a large block in any computation system and hence it becomes a source of large delay and high power dissipation system. in any digital hardware, the widely followed algorithms are array multiplication and booth algorithm but a large propagation delay is occurred in these cases and hence various algorithms have been designed to obtain high speed and low power efficient multipliers and hence came the concept of vedic multiplication. The detailed information about the Vedic multiplier and its rule has been explained in the further sub sections.

2.8.1 Vedic Multiplier

The word "Vedic" is a Sanskrit word. In Sanskrit the word, 'Veda' means store house of all knowledge. Vedic mathematics is based on 16 Sutras (or aphorisms).These sutras deal with various branches of mathematics. In Vedic mathematics, methods adapted for basic arithmetic are simple and powerful [38, 39]. In Vedic mathematics, Urdhva Tiryagbhyam sutra is used generally for high speed multiplication. Vedic multiplier is very much advantageous in case of bigger multiplication which would consume more time in a conventional multiplier.

2.8.2 Details of Urdhva-Tiryagbhyam Method of Multiplication

Urdhva Tiryagbhyam sutra is a sutra for multiplication in the ancient Indian Vedic mathematics. This is based on the generation of all partial products by means of concurrent addition of the partial product. Urdhva Tiryagbhyam generates partial products and their summation in parallel and not sequential with more number of steps as done in conventional multiplication.

Vedic mathematics reduces the typical calculations in conventional mathematics into very simple one. This sutra works on the principle on which a human brain works, and thus follows the natural principles to generate Vedic formulae.

Due to the implementation of natural principle, the Vedic formulae are more efficient with respect to speed than their counterpart. The multiplication sutra namely the Urdhva

Tiryagbhyam is used for all high speed multiplication. The word “Urdhva” means “Vertically” and “Tiryagbhyam” means “Crosswise” in Sanskrit.

Therefore, this algorithm uses the concept of vertical and crosswise multiplication and addition to get the partial products. To illustrate this multiplication scheme, the multiplication of two decimal numbers is chosen as shown in Fig.13.

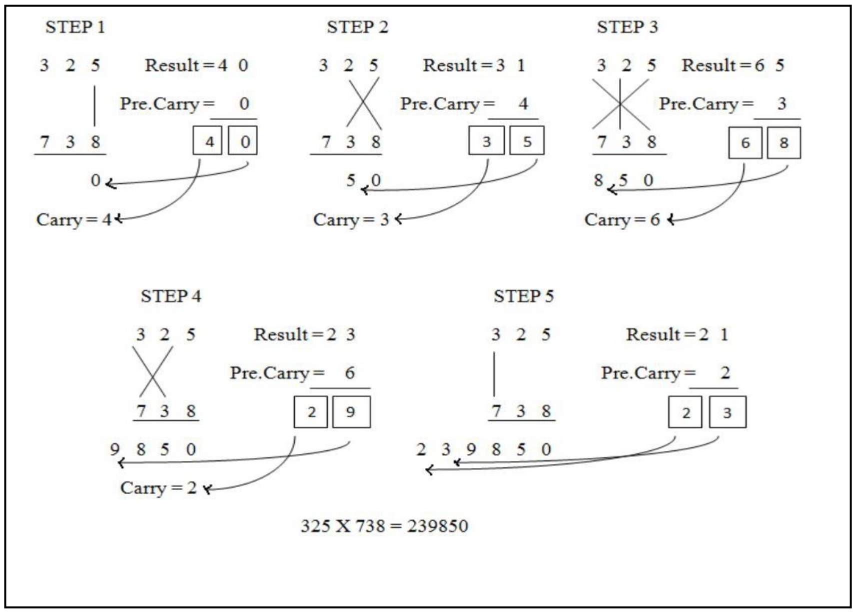

Fig.13 Multiplication of two decimal numbers using Vedic mathematics.

2.8.3 Algorithm

Fig. 14 shows the line diagram for the multiplication. The line diagram demonstrates the procedure for multiplication of 2, 3 and 4 bits. In the line diagram, the dots represent bit “0” or “1”. From the line diagram of 2 bit multiplication, it is seen that the digits on both sides of the line are multiplied and added with the carry from the previous step [40]. The result generates one of the bits of the result and a carry. In the next step, this carry is added and hence the process goes on. The same procedure is followed for more number of bits. The line diagram shows the methodology adapted for 3 and 4 bits [41].

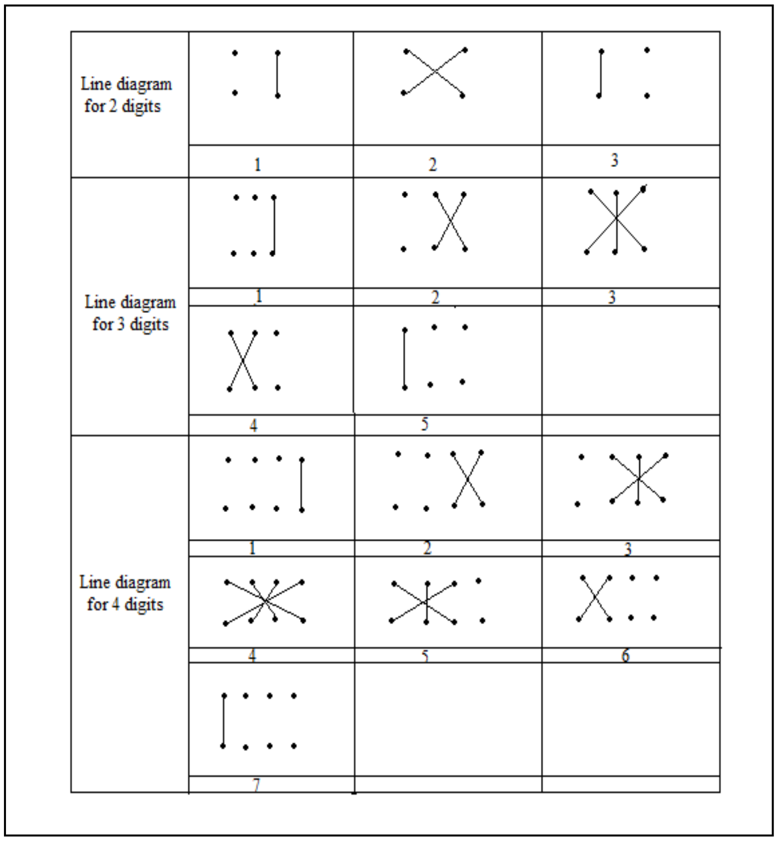
Line diagram for 2 digits
1
2
3
Line diagram for 3 digits
1
2
3
4
5
Line diagram for 4 digits
1
2
3
4
5
6
7

Fig.14 Line Diagram for Vedic Multiplication of 2, 3 and 4 digits.

2.9 Urdhva Multiplier Hardware Architecture

The 'Urdhva Tiryagbhyam' algorithm can be implemented in the same way for binary number system as decimal number system. As an example, a 4x4 Vedic multiplier hardware implementation is explained.

The 4x4 bit Vedic multiplier module is implemented using four 2x2 bit Vedic multiplier modules. Let’s examine 4x4 multiplication say a=a3a2a1a0 and b=b3b2b1b0. The output for the multiplication result is s7s6s5s4 s3s2s1s0. Divide a and b into two parts say a3a2 & a1a0 for a and b3b2 & b1b0 for b. Then two bits are taken as inputs to the 2bit multiplier block which uses the Vedic mathematic fundamentals.

The structure for multiplication will be as shown in Fig. 15. Each block, as shown is a 2x2 bit Vedic multiplier. The inputs to the first multiplier are a1a0 and b1b0. The last block is a 2x2 bit multiplier with inputs a3a2 and b3b2.

The middle one shows two 2x2 bit multipliers with inputs a3a2 & b1b0 and a1a0 & b3b2. The final result of multiplication is of 8 bits, s7s6s5s4 s3s2s1s0. To get final product (s7s6s5s4 s3s2s1s0) four 2x2 bit Vedic multiplier and three 4-bit Ripple-Carry (RC) adders are required.

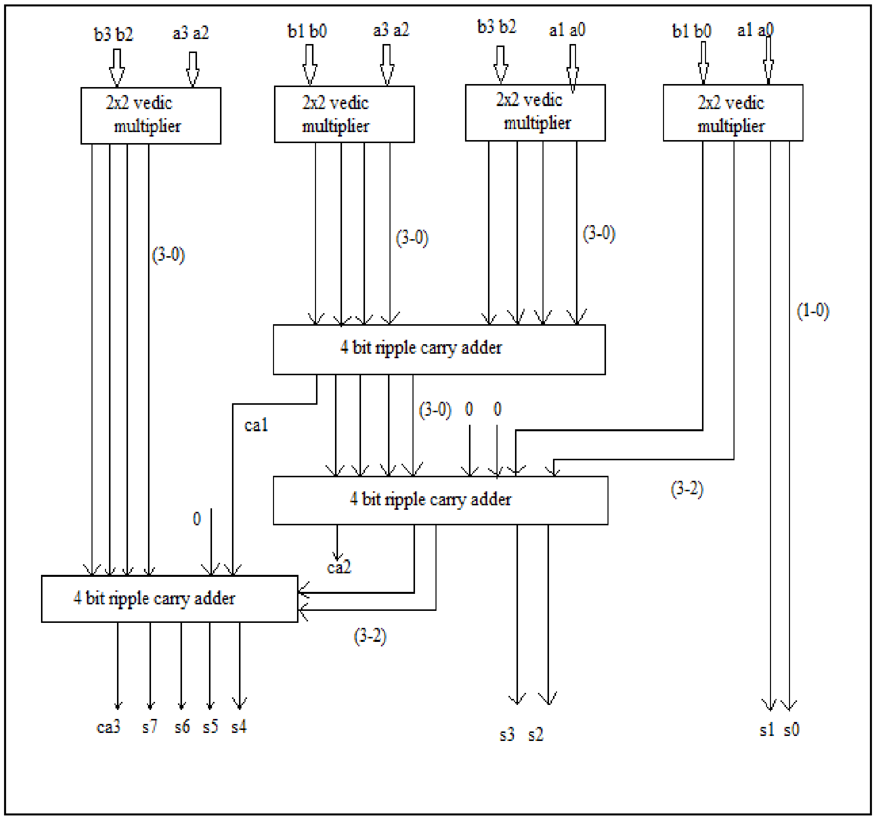

Fig.15 Block Diagram of 4x4 bit Vedic Multiplier.

The figure above describes the block diagram of 4 bit Vedic multiplier and it is clearly shown that the 4 bit multiplier is implemented by using four 2 bit Vedic multipliers. Hence, the Fig.16 below shows the RTL schematic of 2 bit multiplier used in designing of 4 bit multiplier.

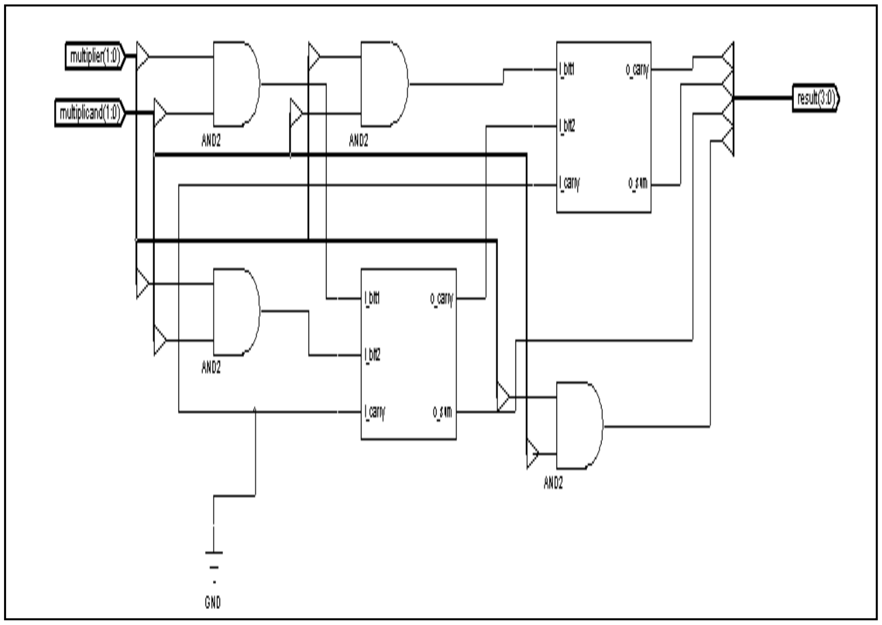

Fig.19 RTL schematic of 2 bit multiplier.

CHAPTER 3

NEURONAL LOGIC

The brain is composed of Boolean entities functioning as threshold units. These simplified units constitute pure and reliable logic-gates (e.g., AND, XOR), similar to the logic at the core of computers. Generalization of this simplified Boolean framework to include unreliable elements has emerged in 1956 by the innovative work of John von Neumann. The Von Neumann concepts as well as the earlier pioneering work of Claude Shannon to simplify Boolean circuits are at the cornerstone of the present day's computational paradigm [42].

Using the McCulloch-Pitts model we can model logic functions. As shown below, Fig.17 shows and describes the architecture for three logic functions – AND, OR & AND-NOT.

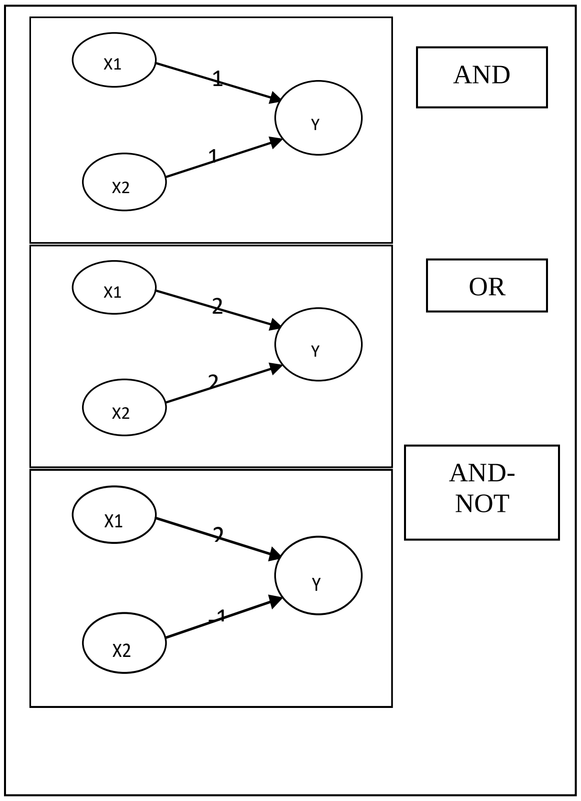

Fig.20 Neuronal logic gates.

3.1 Neuronal Logic Gates

The truth tables for each function are also shown in Table II.

TABLE II. TRUTH TABLE FOR AND, OR , AND-NOT GATE.

INPUTS		OUTPUTS		
X_1	X_2	AND	OR	AND-NOT
0	0	0	0	0
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

3.1.1 AND gate

X_1 and X_2 are the inputs connected to the same neuron. For modeling the AND function the threshold on Y and the weights can be set according to the following conditions.

For case 1(when$x_1 = x_2 = 0$)

$$\omega_1 x_1 + \omega_2 x_2 - \theta < 0 \text{ i.e } \theta > 0 \quad (9)$$

Similarly for other cases

$$\omega_1 x_1 + \omega_2 x_2 - \theta < 0 \text{ i.e } \omega_2 < \theta \quad (10)$$

$$\omega_1 x_1 + \omega_2 x_2 - \theta < 0 \text{ i.e } \omega_1 < \theta \quad (11)$$

$$\omega_1 x_1 + \omega_2 x_2 - \theta \geq 0 \text{ i.e } \omega_1 + \omega_2 \geq \theta \quad (12)$$

where ω, x and θ are the weight, input and threshold respectively.

3.1.2 OR gate

OR function is similar to the AND function except the equations will be slightly modified according to the truth table and hence the threshold and weights can be correctly chosen as per equations

For case 1(when$x_1 = x_2 = 0$)

$$\omega_1 x_1 + \omega_2 x_2 - \theta < 0 \text{ i.e } \theta > 0 \quad (13)$$

Similarly for other cases

$$\omega_1 x_1 + \omega_2 x_2 - \theta \geq 0 \text{ i.e } \omega_2 > \theta \quad (14)$$

$\omega_1 x_1 + \omega_2 x_2 - \theta \geq 0$ i.e $\omega_1 > \theta$ (15)

$\omega_1 x_1 + \omega_2 x_2 - \theta \geq 0$ i.e $\omega_1 + \omega_2 \geq \theta$ (16)

where ω, x and θ are the weight, input and threshold respectively

3.1.3 AND-NOT gate

The AND-NOT function is not symmetric, in that an input of 1, 0 will be treated in a different way to an input of 0, 1. It is clear from the truth table that 'true' (value of one) is returned when the first input is true and the second input is false. Again, the threshold and weights can be chosen according to the desired equations obtained from truth table.

For case 1(when$x_1 = x_2 = 0$)

$\omega_1 x_1 + \omega_2 x_2 - \theta < 0$ i.e $\theta > 0$ (17)

Similarly for other cases

$\omega_1 x_1 + \omega_2 x_2 - \theta < 0$ i.e $\omega_2 < \theta$ (18)

$\omega_1 x_1 + \omega_2 x_2 - \theta \geq 0$ i.e $\omega_1 \geq \theta$ (19)

$\omega_1 x_1 + \omega_2 x_2 - \theta < 0$ i.e $\omega_1 + \omega_2 < \theta$ (20)

where ω, x and θ are the weight, input and threshold respectively.

3.2 Limitation of simple Perceptron

As we have seen before that AND, OR, and NOT gate can ne easily implemnted using a simple perceptron model. But what if we want to design a XOR gate using the same principle.

TABLE III. TRUTH TABLE FOR XOR GATE.

INPUTS		**OUTPUT**
X1	X2	
0	0	0
0	1	1
1	0	1
1	1	0

Table III shows the truth table of XOR gate. For all the four cased of inputs different equations can be derived which will decide the threshold value and the different values of weights that will satisfy the logical XOR function. Hence, from the truth table of XOR gate the equations obtained will be:

For case 1(when$x_1 = x_2 = 0$)

$\omega_1 x_1 + \omega_2 x_2 - \theta < 0$ i.e $\theta > 0$ (21)

Similarly for other cases

$\omega_1 x_1 + \omega_2 x_2 - \theta < 0$ i.e $\omega_2 < \theta$ (22)

$\omega_1 x_1 + \omega_2 x_2 - \theta \geq 0$ i.e $\omega_1 \geq \theta$ (23)

$\omega_1 x_1 + \omega_2 x_2 - \theta < 0$ i.e $\omega_1 + \omega_2 < \theta$ (24)

where ω, x and θ are the weight, input and threshold respectively.

From the last three equations we can see that the equation (22) and (23) are not compatible with the equation (24). No value of threshold and weights will satisfy the all the above equations. Hence, no solution exists in this case and XOR gate cannot be realized using simple Perceptron. Hence to realize such gates more complex networks are needed in which it would not be possible to calculate the weights and threshold by hand as in the case of AND, OR and NOT gate. Hence a multilayer network will be used to realize such problems. As explained in previous chapters that such networks consists of an input layer, hidden layer and an output layer and it involves some learning mechanism to calculate the required parameters.

CHAPTER 4

PROPOSED HARDWARE ARCHITECTURE OF NEURAL NETWORK IMPLEMENTED USING VEDIC MULTIPLIER

The proposed design consists of a neural network design consisting of input layer and output layer, activation function design and high speed multiplier design which is the main component of the proposed design. To get a high performance neural network the proposed design utilizes vedic multiplier due to its obvious advantages explained in previous chapters.

4.1 Proposed hardware architecture of Vedic neuron.

The main element of any neural network starts with the designing of a neuron. It processes data in three steps:

- The inputs are multiplied with the weights i.e. weighing of input values.
- All the multiplied values are summed by a serial accumulation/adder.
- Filtering by the activation function

In the proposed design the standard multiplier has been replaced with the Vedic multiplier to enhance the speed, area and power of any neural network architecture. It is done because processing in neural network involves a lot of multiplication at each step as soon as input is applied. Hence, to avoid the delays incurred due to the standard multipliers, Vedic multiplier has been used. A four bit Vedic multiplier has been designed and is made using four 2 bit Vedic multipliers and three ripple carry adders. It is based on Urdhva Tiryagbhyam sutra which is based on vertical and crosswise algorithm. All the multiplier blocks are designed in VHDL and results are verified through modelsim. Analysis shows that it is more efficient than the standard one.

Also, activation function forms an important part of neuron. Based on the choice of activation function, various applications can be designed. In the proposed work we have utilized the most common and simplest activation function, i.e. threshold function. The threshold value can be set as per the desired application.

The threshold function is given as

$$\emptyset(v) = \begin{cases} 1 \; if \; v \geq 0 \\ 0 \; if \; v < 0 \end{cases} \qquad (25)$$

where v is the threshold value.

The neuron will only be fired when the when the summation of the weighted inputs will be greater than the specified value. If it is less than the set value than the output remains zero or we say that neuron is not fired.

The proposed hardware is almost similar to a standard non-linear neuron (McCulloch Pitts model) except for the multiplier. Here the standard multiplier is replaced with a Vedic multiplier which multiplies the inputs with weights. For evaluation purpose the weights were randomly fixed. Fig. 18 shows the proposed architecture of a single neuron which has been designed using Vedic multiplier instead of standard multiplier, to achieve higher speed in designing neural nets.

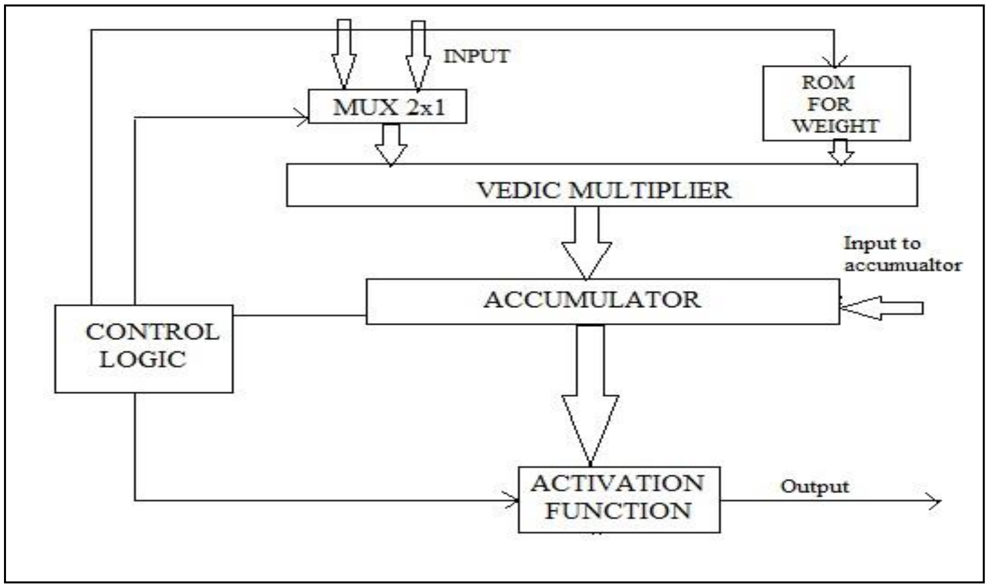

Fig.21 Proposed Architecture.

The proposed architecture has the advantages of high speed, which can be of use in various applications the involves neural network computing.

The functioning of the proposed architecture as the three basic neuronal logic gates i.e. AND, OR and NOT has also been shown and the results show a drastic improvement in the efficiency of neuronal logic gates.

4.2 Flow chart

The flow chart shown below explains the stepwise working of the proposed design and hence the possibility when a neuron is fired.

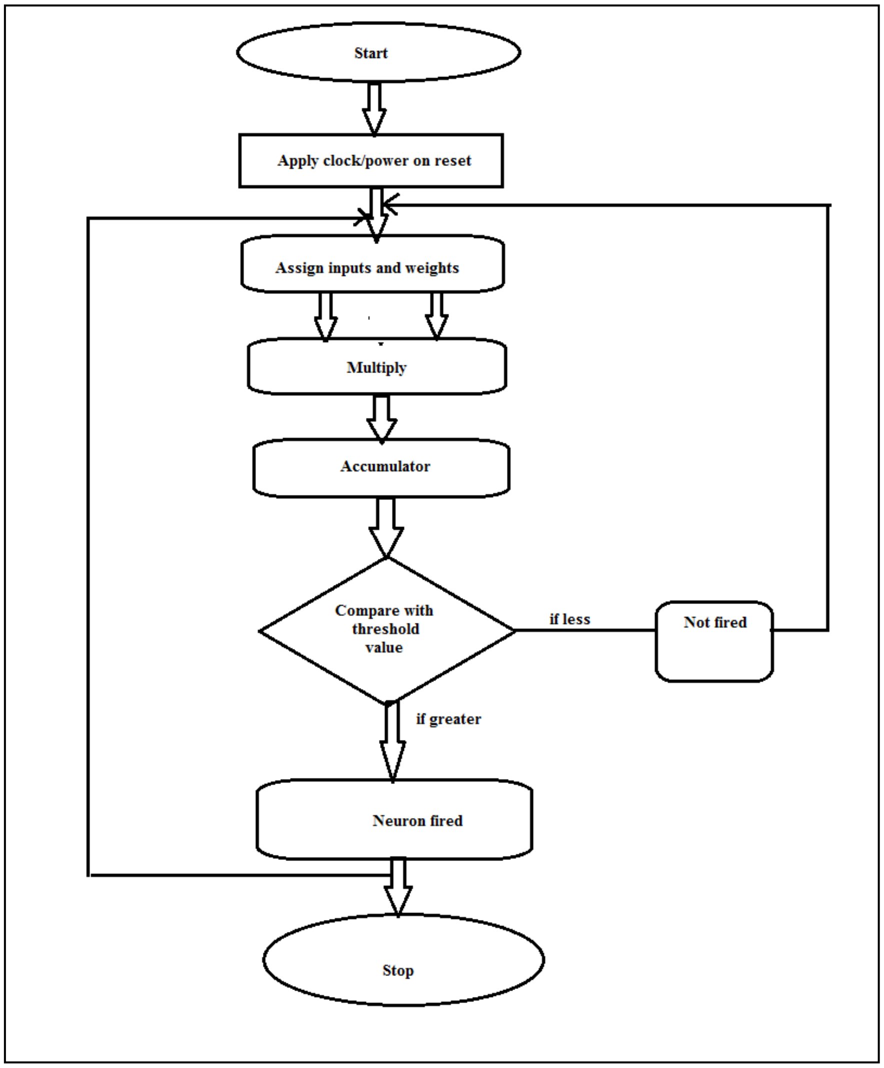
Start
Apply clock/power on reset
Assign inputs and weights
Multiply
Accumulator
Compare with threshold value
if less
Not fired
if greater
Neuron fired
Stop

Fig.22 Flow chart representing the steps followed in the proposed design.

The process starts with when the clock and power on reset are applied. As soon as the inputs are available, they are being multiplied with the desired weights. The result from the multplier is added with the other results and the output of the adder/accumualtor is provided to the activation function (here threshold logic). Now, if the output of the adder/accumulator is greater than the threshold value than the neuron will be fired else it is not and the process will repeat.

Similarly for the verification of neuronal gates appropriate value of threshold will be decided and hence the desired inputs will be applied at the input of multiplier and hence the required results will be obtained.

CHAPTER 5

SIMULATION RESULTS

For the purpose of evaluation, performance of a single neuron (with fixed weights) with standard multiplier and Vedic multiplier were compared. The hardware implementation was carried out using VHDL. For implementation, testing and simulation purpose, Xilinx ISE 6.1 with ModelSim simulation tools were used. Spartan II family was been chosen as target device. The simulation and synthesis results are as shown:

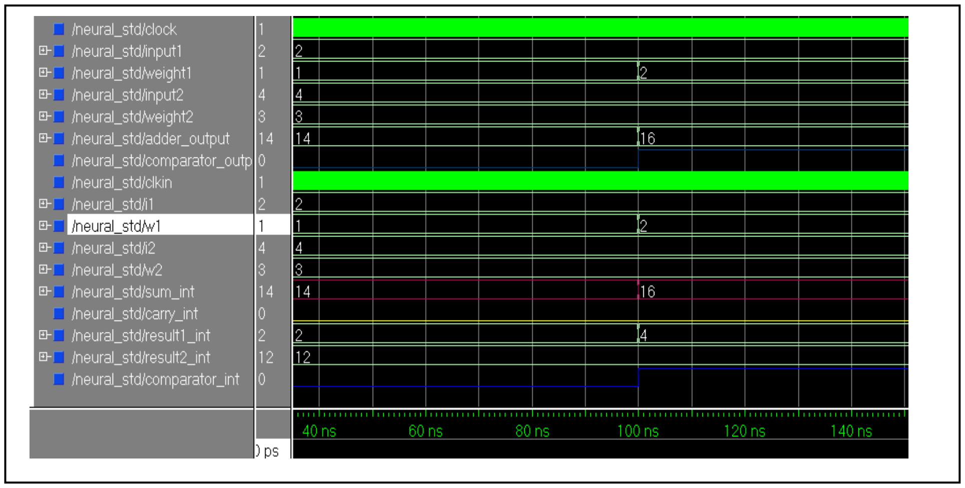

Fig.23 Simulation results of a neuron using standard multiplier.

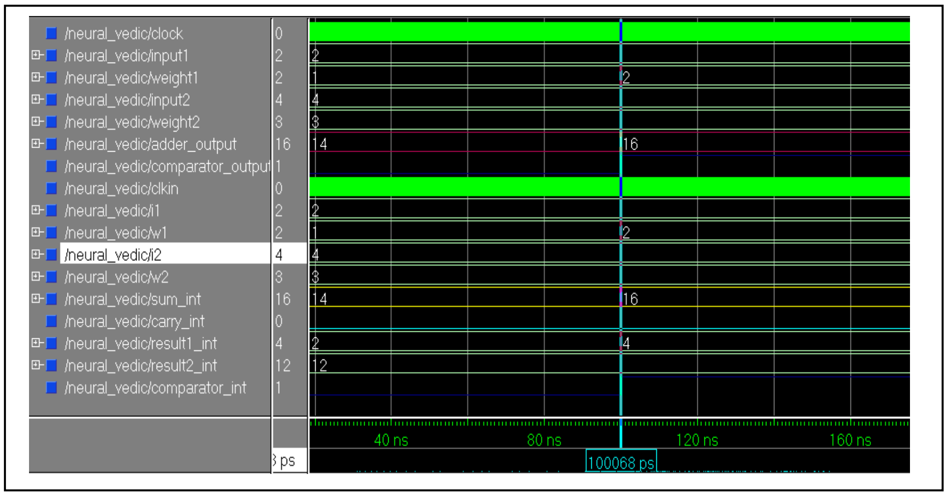

Fig.24 Simulation results of a neuron using Vedic multiplier.

When any input is applied, it is multiplied with the desired weight (here chosen randomly) and sent to the adder which also contains the output of multiplier with different inputs. The adder adds the outputs and the result is send to the activation function which is threshold function here. A threshold of 15 is selected here and hence it can be clearly seen from Fig.20 and 21 that when the adder output is 14 the neuron is not fired and when the output is 16 the comparator output goes high and hence the neuron is said to be fired.

5.1 Simulation Results of Neuronal gates

The simulation results of all the four cases of neuronal AND gate and OR gate using conventional and Vedic multiplier are shown in Fig.22-25.

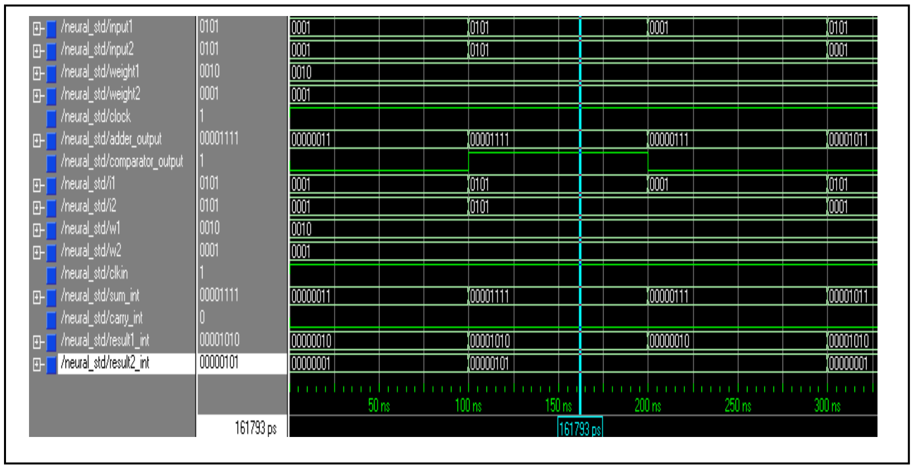

Fig.25 Conventional Neuronal AND gate.

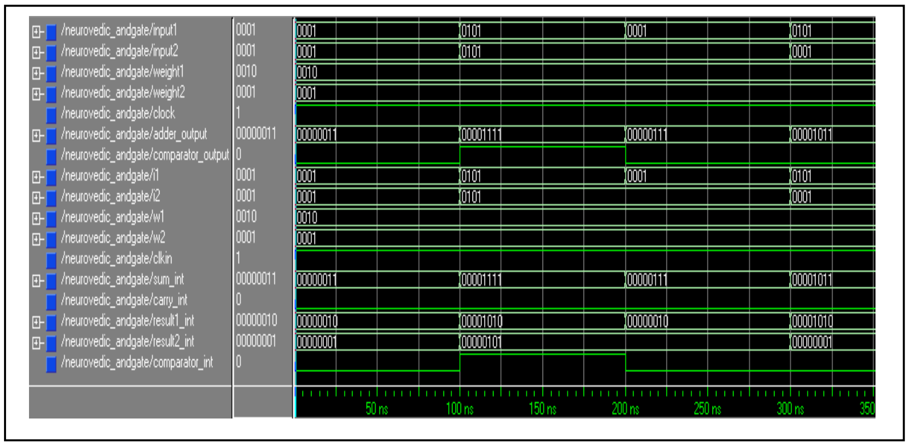
/neurovedic_andgate/input1 0001
/neurovedic_andgate/input2 0001
/neurovedic_andgate/weight1 0010
/neurovedic_andgate/weight2 0001
/neurovedic_andgate/clock 1
/neurovedic_andgate/adder_output 00000011
/neurovedic_andgate/comparator_output 0
/neurovedic_andgate/i1 0001
/neurovedic_andgate/i2 0001
/neurovedic_andgate/w1 0010
/neurovedic_andgate/w2 0001
/neurovedic_andgate/clkin 1
/neurovedic_andgate/sum_int 00000011
/neurovedic_andgate/carry_int 0
/neurovedic_andgate/result1_int 00000010
/neurovedic_andgate/result2_int 00000001
/neurovedic_andgate/comparator_int 0
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350

Fig.26 Vedic neuronal AND gate.

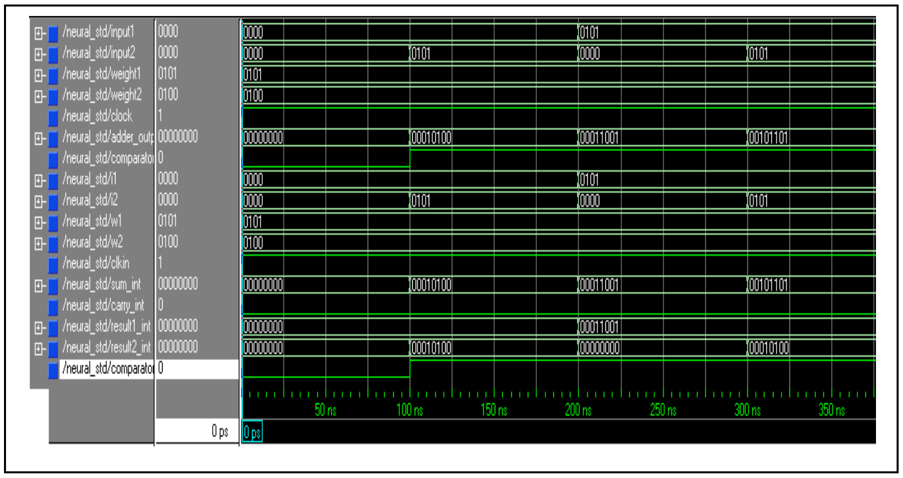

Fig.27 Conventional OR gate.

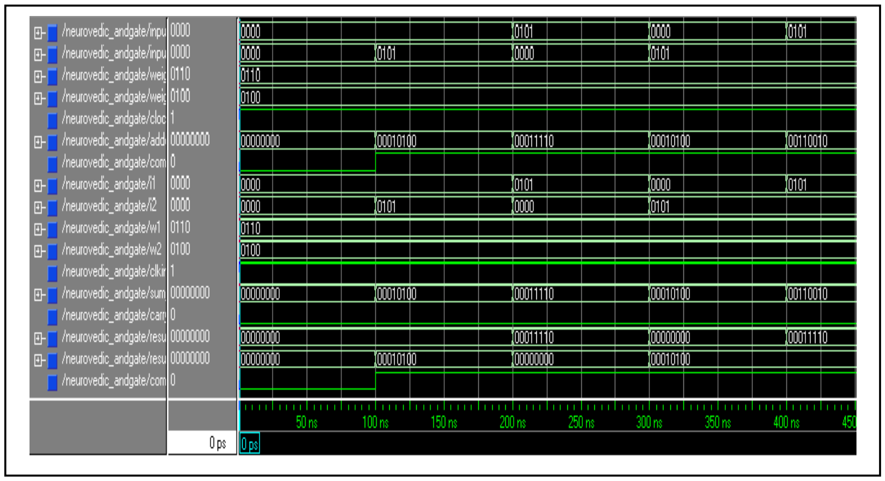

Fig.28 Vedic neuronal OR gate.

The AND and OR gate are verified considering all the four cases and deciding the threshold value according to the equations described in chapter 4. In AND gate, for simplicity logical 0 is taken as 1V and logical 1 is taken as 5V. And the threshold value is taken as 12V as per the desired equations. Similarly the truth table for OR gate is verified and the results shows that the gates designed using neuronal logic are much faster as compared to the standard one.

TABLE IV. SYNTHESIS RESULT OF NEURON WITH STANDARD MULTIPLIER

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	72	192	37%
Number of 4 input	125	384	32%
Number of bonded	25	90	27%

TABLE V. SYNTHESIS RESULT OF NEURON WITH VEDIC MULTIPLIER

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	40	192	20%
Number of 4 input	70	384	18%
Number of bonded	25	90	27%

TABLE VI. DELAY COMPARISON OF NEURON

FPGA device package: xc2s15-6-cs144	**Delay**	**Memory(Kb)**
Using standard multiplier	32.969	72772
Using Vedic multiplier	25.675	73668

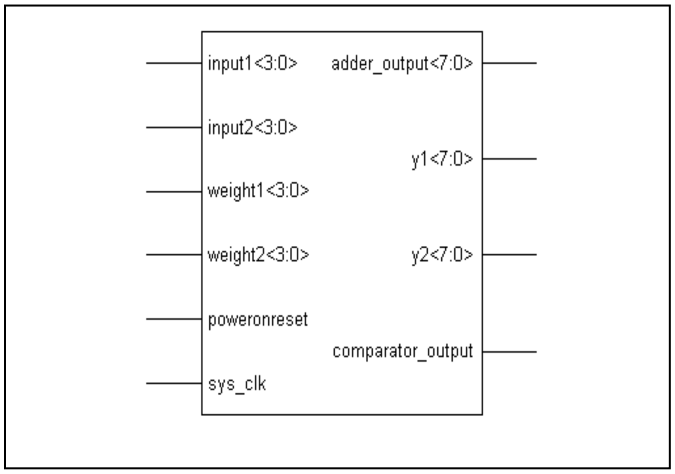

Fig.29 Top level design of Vedic neuron.

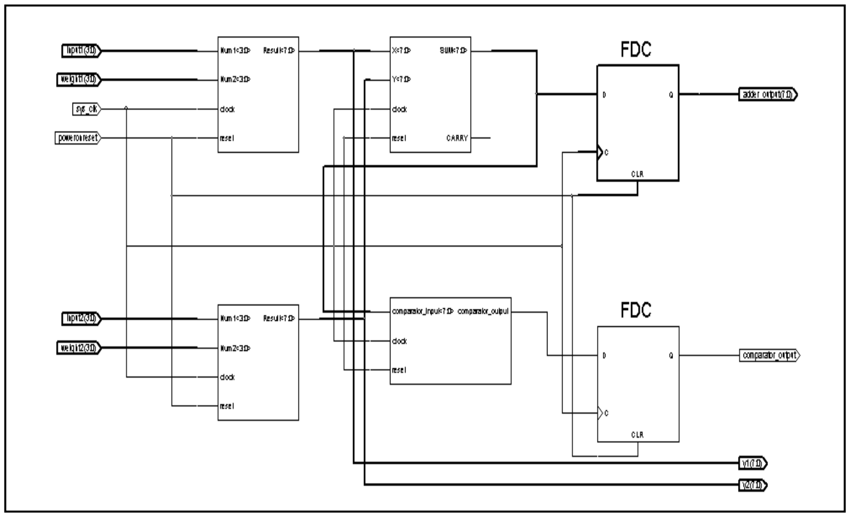

Fig.30 RTL schematic of Vedic neuron.

5.2 Comparison of standard neural network with Vedic neural network.

Table IV & V shows the logic utilization of a neuronal gate with conventional and Vedic multiplier. Table VI compares the result with respect to speed of neuronal gates realized using conventional and Vedic multiplier. Fig.26 and 27 shows the black box view and RTL schematic of the neural network designed using Vedic mathematics.

The simulation results indicate that the delay of logic gate using conventional multiplier is 33ns while this delay reduces to 25ns with the use of Vedic multiplier. As compared to a standard multiplier, the use of Vedic multiplier reduces the logic gate realization time by approximately 25%. This decrease in the neuronal gate's latency can be of great use in applications where speed is very critical.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

The technologies like ANN & Vedic mathematics (multiplier) are the two important and promising technologies which can increase the processing speed. In this work, a novel scheme to combine these two technologies to achieve a high performance neuron or ANN has been proposed. A single neuron is implemented using Vedic mathematics (Vedic neuron) in VHDL. The results of standard neuron and Vedic neuron are compared. Hence, the comparison result indicates that the speed of Vedic neuron is better than standard neuron.

The high speed of neural network is well proven. This has led to its applications in numerous fields like image compression, security, and medical applications. The speed of a neuron can further be increased by the use of Vedic multiplier. In the presented work, a novel approach has been proposed to achieve a high performance neuronal logic gates. The proposed approach combines these two technologies. Thorough result and speed analysis has been performed, that indicates approximately 25% reduction in the processing time of a single gate. Use of Vedic multiplier for the realization of gates and complete neural network will reduce the processing time multifold. Neuronal logic gates implemented using Vedic neuron will find its applications in various fields where computational speed is the main concern.

6.2 Future Work

This research can be extended by replacing Vedic adders in place of standard adders in neurons, which would in all means increase the computation speed. Since, neurons are the basic building blocks of Neurocomputers, the speed of Neurocomputers developed using Vedic neurons would be better than the standard Neurocomputers. Also, the concept of neural networks and fuzzy logic is being used in many applications today. Hence, the combination of these two technologies i.e. ANN and Vedic mathematics in turn will enhance all the applications that utilize the concept of ANN.

REFERENCES

[1] Girish Kumar Jha, "Artificial neural networks and its applications", Indian Agricultural Research Institute, PUSA, New Delhi, 2007.

[2] Huqqani A.A, Schikuta E, and Mann E, "Parallelized neural networks as a service in neural networks", (IJCNN) International Joint Conference on Neural Networks, IEEE, 2014, pp. 2282-2289.

[3] T. Theocharides, G. Link, N. Vijaykrishnan, M. J. Irwin, and V. Srikantam, "A generic reconfigurable neural network architecture implemented as a network on Chip", Proc. of the IEEE International System on Chip Conference, September 2004, pp.191-194.

[4] Omondi, Amos R., and Jagath Chandana Rajapakse, eds, "FPGA implementations of neural networks", Vol. 365, New York, USA, Springer, 2006.

[5] Janusz Starzyk, and Jing Pang, "Evolvable binary artificial neural network for data classification", (PDPTA) Parallel and Distributed Processing Techniques and Applications, Distributed & Parallel Computing, 1995-2013.

[6] Chun-Hsien Chen and Vasant Honavar, "A Neural-Network Architecture for Syntax Analysis", IEEE Transactions on Neural Networks, vol. 10, no. 1,January 1999, pp. 94-114.

[7] Eldredge, James G., and Brad L. Hutchings, "RRANN: a hardware implementation of the backpropagation algorithm using reconfigurable FPGAs", International Conference on World Congress on Computational Intelligence In Neural Network., IEEE, 1994, vol. 4, pp. 2097-2102.

[8] Y. Hirai, "Hardware implementations of neural networks in Japan", Neurocomputing, Elsevier, 1993, vol.5, pp.3–16.

[9] Sundararajan, N., and P. Saratchandran, "Parallel architectures for artificial neural networks: paradigms and implementations", IEEE Computer Society Press, 1998.

[10] Hammerstrom, Dan, "A digital VLSI architecture for real-world applications", an introduction to neural and electronic networks, 1995, pp. 335-358.

[11] Rückert, Ulrich, "ULSI architectures for artificial neural networks", IEEE Micro 22, no. 3, 2002, pp. 10-19.

[12] Buddefeld, Jurgen, and Karl E. Grosspietsch, "Intelligent-memory architecture for artificial neural networks", Micro, IEEE 22, no. 3, 2002, pp. 32-40.

[13] Danese, Giovanni, Francesco Leporati, and Stefano Ramat, "A parallel neural processor for real-time applications", IEEE Micro 22, no. 3, 2002, pp. 20-31.

[14] H.Guilt, "Fully iterative fast array for binary multiplication", Electronics Letters, vol. 5, 1969, pp. 263.

[15] Ramalatha, M Dayalan, K D Dharani, P Priya, and S Deborah, "High speed energy efficient ALU design using Vedic multiplication techniques", ICACTEA, Jul 15-17, 2009, pp. 600-603.

[16] Kunchigi, Vaijyanath, Linganagouda Kulkarni, and Subhash Kulkarni, "High speed and area efficient vedic multiplier", (ICDCS) International Conference on Devices, Circuits and Systems, IEEE, 2012, pp. 360-364.

[17] Y.bansal, C.Madhu, and P.Kaur, "High speed Vedic multiplier designs: A review," Proceedings of IEEE, Recent Advances in Engineering and Computational Sciences (RAECS), 06- 08 March 2014.

[18] D.Jaina, K.Sethi, and R. Panda, "Vedic Mathematics based Multiply Accumulate Unit", International Conference on computational Intelligence and Communication System, IEEE, 2011.

[19] Veluswami, Anand, Michel S. Nakhla, and Qi-Jun Zhang, “The application of neural networks to EM-based simulation and optimization of interconnects in high-speed VLSI circuits”, IEEE Transactions on Microwave Theory and Techniques, 45, no. 5, 1997, pp. 712-723.

[20] Vanajakshi, Lelitha, and Laurence R. Rilett, “A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed”, In Intelligent Vehicles Symposium, IEEE, 2004, pp. 194-199.

[21] A.Veluswami, A.hafid Zaabab, Qi-jun Zhang and Michel S. Nakhla, “The appliaction of neural networks to high speed interconnect simulation”, IEEE, 1995

[22] Huddar, Sushma R., Sudhir Rao Rupanagudi, M. Kalpana, and Surabhi Mohan, “Novel high speed vedic mathematics multiplier using compressors.”, 2013 International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing, IEEE, 2013, pp. 465-469.

[23] Jaina, Devika, Kabiraj Sethi, and Rutuparna Panda. "Vedic mathematics based multiply accumulate unit." In Computational Intelligence and Communication Networks (CICN), 2011 International Conference on, pp. 754-757. IEEE, 2011.

[24] Fredkin, Edward, and Tommaso Toffoli, “Conservative logic”, Springer London, 2002.

[25] Haykin, S., “Neural networks a comprehensive foundation”, Prentice Hall Publishing, Vol.1, 1999, 2nd edition, pp 6-7.

[26] Bose, N. K, and Liang P, “Neural network fundamentals with graphs, algorithms, and applications”, McGraw-Hill, 1996.

[27] Purushottam D. Chidgupkar, and Mangesh T. Karad, “The implementation of Vedic algorithms in digital signal processing”, UNESCO International Centre for Engineering Education (UICEE), Australia, vol.8, no.2, 2004.

[28] H. Thapliyal, and H.R Arbania, “A Time-Area-Power Efficient Multiplier and Square Architecture Based On Ancient Indian Vedic Mathematics”, Proceedings of the 2004 International Conference on VLSI, June 2004, pp. 434-439.

[29] Miradi, Maryam, “Development of intelligent models for ravelling using neural network”, International Conference on In Systems, Man and Cybernetics, IEEE, vol. 4, 2004, pp. 3599-3606.

[30] D.S. Broomhead and D.lowe, “Mutivariate functional interpolation and adaptive networks”, Complex systems, 1988, pp.321-355.

[31] T.Nordstrom and B.Svensson, “Using and designing massively parallel computers for artificial neural networks”, Journal of Parallel & Distributed Computing, 1991, pp.260-285.

[32] J. Misra,and I. Saha, “Artificial neural networks in hardware: A survey of two decades of progress”, Elsevier, Neurocomputing vol.74, 2010, pp. 239-255.

[33] A. Tisan, M.N. Cirstea, S. Oniga ,and A. Buchman, “ Artificial olfaction system with hardware on-chip learning neural networks”, Proc. of IEEE Int. Conference on Optimisation of Electrical and Electronic Equipment ,2010,pp. 884-889.

[34] W. Kurdthongmee, “A novel hardware-oriented Kohonen SOM image compression algorithm and its FPGA implementation”, Journal of Systems Architecture, vol.54, 2008, pp.983–994.

[35] A. Gomperts, A. Ukil, and F. Zurfluh, “Development and implementation of parameterized FPGA-based general purpose neural networks for online applications”, IEEE Transactions on Industrial Informatics, 2011, pp 78-89.

[36] Cheng-Jian Lin and Hung-Ming Tsai, “FPGA implementation of a wavelet neural network with particle swarm optimization learning”, Elsevier, Mathematical and Computer Modeling, vol. 47,2008, pp 982–996.

[37] J. M. Granado, M. A. Vega, R. Pérez, J. M. Sánchez,and J. A. Gómez, “Using FPGAs to Implement Artificial Neural Networks”,13th International Conference on Electronics, Circuits and Systems, ICECS, IEEE, 2006, pp 934-937.

[38] Harpreet Singh Dhillon, and Abhijit Mitra, “A reduced bit multiplication algorithm for digital arithmetic,” International Journal of Computational and Mathematical Sciences, no.2, 2008.

[39] Shamim Akhter, “VHDL implementation of fast NXN multiplier based on Vedic mathematics”, 18th European Conference on Circuit Theory and Design, ECCTD, IEEE, 2007, pp. 472-475.

[40] Vinay, “Analysis, Verification and FPGA implementation of Vedic multiplier with BIST capability”, PhD thesis, Thapar University, 2009.

[41] Anju, and V.K.Agrawal, “FPGA implementation of low power and high speed Vedic multiplier using Vedic mathematics”. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), volume 2, issue 5, May – Jun. 2013, pp. 51-57.

[42] Goldental, Amir, Shoshana Guberman, Roni Vardi, and Ido Kanter, “A computational paradigm for dynamic logic-gates in neuronal activity”, Frontiers in computational neuroscience, 8, 2014.

APPENDIX

Publication from the current work

- High speed neuron implementation using Vedic mathematics.

  (Accepted and presented at IEEE supported CCEEDS international conference.)